Olli Tuominen

Designing Principles for Metropolia Education Cloud Platform

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

10 April 2017

Helsinki Metropolia University of Applied Sciences

Author Title Number of Pages Date	Olli Tuominen Designing principles for Metropolia Education Cloud Platform 35 pages + 2 appendices 10 April 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Data Networks
Instructor	Harri Ahola, Senior Lecturer

The goal of the project was to gain experience from OpenStack and SUSE OpenStack Cloud technologies. SUSE OpenStack Cloud was chosen to be used in the Metropolia Education Cloud Platform. The secondary goal of the project was to design a highly available OpenStack cloud platform, which would be implemented as the Metropolia Education Cloud Platform.

The Metropolia Education Cloud Platform requires certain elements that had to be taken into account in the project. These requirements were qualified and validated so that they could be met with the chosen technology.

As a result of this project, a reference architecture for the Metropolia Education Cloud Platform with SUSE OpenStack Cloud was designed. Moreover, design principles to design a highly available OpenStack Cloud were created.

OpenStack was found to be the technology that will be used for the next generation's software-defined data centers. The design principles created in the project can be used to design any other highly available OpenStack cloud platform. However, the overall success of designing depends on the fact that the scope of a project is planned well.

Keywords	cloud computing, OpenStack, SUSE, private cloud

Tekijä Otsikko Sivumäärä Aika	Olli Tuominen Metropolian opetuspilvialustan suunnitteluperiaatteet 35 sivua + 2 liitettä 10.4.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja	Lehtori Harri Ahola

Insinöörityön tarkoituksena oli kartuttaa kokemuksia OpenStack- ja SUSE OpenStack Cloud -teknologiasta. SUSE OpenStack Cloud on valittu käytettäväksi Metropolia-opetuspilvi -ympäristössä. Insinöörityön tavoitteena oli myös suunnitella korkeasti käytettävissä oleva OpenStack-pilvialusta, joka otettaisiin käyttöön Metropolia-opetuspilvenä.

Metropolia-opetuspilvi edellyttää tiettyjen vaatimusten täyttymistä, jotka on otettava huomioon suunnittelussa. Insinöörityössä vaatimukset määriteltiin ja validoitiin, jotta ne voidaan täyttää valitulla teknologialla. Vaatimuksena oli esimerkiksi, että opetuspilvi mahdollistaa itsepalvelukäyttöliittymän kautta tilattujen palveluiden tilaamisen ja käytön.

Työssä suunniteltiin Metropolian opetuspilvialustan arkkitehtuuri käyttäen SUSE OpenStack Cloud -teknologiaa. Lisäksi luotiin suunnittelun periaatteet, joiden avulla voidaan suunnitella korkeasti käytettävissä oleva OpenStack-pilviympäristö. Suunnittelun periaatteista tärkeimpänä voidaan mainita niin kutsutun single point of failure -pisteen välttämistä. Tällöin yksittäinen rikkoutunut piste ei lopeta OpenStack-pilviympäristön käyttöä.

OpenStackin havaittiin olevan IT-teknologia, jota tullaan käyttämään seuraavan sukupolven ohjelmisto-ohjatuissa palvelinkeskuksissa. Insinöörityössä luotujen suunnitteluperiaatteiden avulla voidaan suunnitella muita korkeasti käytettävissä olevia OpenStack-pilvialustoja. Kuitenkin on huomioitava, että suunnittelun onnistuminen riippuu täysin siitä, kuinka hyvin hankkeen tavoitteiden laajuus on määritelty.

Avainsanat	pilvipalvelu, OpenStack, SUSE, yksityinen pilvi

Contents

Abbreviations

API	Application Program Interface
CDN	Content Delivery Network
CPU	Central Processing Unit
CSP	Cloud Service Provider
DHCP	Dynamic Host Control Protocol
DNS	Domain Name System
DRDB	Distributed Replicated Block Device
DVR	Distributed Virtual Router
EPT	Extended Page Table
GRE	Generic Routing Encapsulation
HA	High Availability
IaaS	Infrastructure as a Service
ICT	Information and Communications Technology
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
KSM	Kernel Samepage Merging
KVM	Kernel-based Virtual Machine
ML2	Modular Layer 2
MTU	Maximum Transmission Unit
NFV	Network Function Virtualization
NTP	Network Time Protocol
OCF	Open Cluster Framework
OPNFV	Open Platform for NFV
PaaS	Platform as a Service
PXE	Preboot eXecution Environment
QoS	Quality of Service
SBD	STONITH Block Device
SDN	Software Defined Network
SaaS	Software as a Sevice
SPOF	Single Point of Failure
SSL	Secure Sockets Layer
STONITH	Shoot The Other Node In The Head
TFTP	Trivial File Transfer Protocol
TSL	Transport Security Layer
VIM	Virtualization Infrastructure Manager
VLAN	Virtual LAN
VXLAN	Virtual Extensible LAN
YaST	Yet another Setup Tool

1 Introduction

Helsinki Metropolia University of Applied Sciences (hereinafter *Metropolia*) is one of the Finnish higher-education institutes which has been in the front line of providing cloud technology education in Finland. To meet the increasing demands of providing high-quality education services, and to provide the best of the breed (cloud) learning environment Metropolia's Communications and Network Engineering Department initiated the Metropolia Education Cloud Platform project in 2014. The aim of the project is to create a virtualized data center. Once implemented, the Metropolia Education Cloud Platform can provide cloud services from its data center to Metropolia's faculty and students. It also allows the university to act as a cloud service provider and provide e-learning services outside of the university.

OpenStack technology was chosen to be used as cloud orchestration software in the Metropolia Education Cloud Platform. The driving force to choose OpenStack was the cost savings gained in the software purchases.

The goal of the thesis is to explore OpenStack technology and explain the principles of designing an OpenStack high availability cloud environment, which can be then implemented as Metropolia Education Cloud Platform. The same principles can be used to design any other OpenStack high availability cloud environment. As cloud orchestrator software, SUSE OpenStack Cloud was used as a reference solution.

First, the thesis helps the reader to understand why cloud services are important. It then continues to explain the technology behind OpenStack and SUSE OpenStack Cloud. It is essential to understand the technology before starting to design OpenStack high availability cloud environment based on SUSE OpenStack Cloud technology. After that, design principles are introduced. In chapter 6, a proposed reference architecture, Metropolia Education Cloud Platform, is presented. The thesis ends with a conclusion and recommendations.

2 From Cloud Computing to Cloud Services

2.1 Why Cloud?

Without doubt, cloud computing has been one of the biggest influencers of the 2000 era in the IT sector. Similar analyses have been written more and more by different research companies. However, until now the cloud started to become a reality. So, what has actually changed, and why should the cloud be on everyone's agenda for the next development projects?

Cloud computing has been described as an evolutionary shift [1]. It brings benefits to companies and organizations, which take use of it. It has changed the way companies are building their service offering and the way they are consuming new services. This has been referred to as digitalization, because of the way everything has become digital, and consumerization, because of the way the services are consumed.

Digitalization has helped businesses to generate growth and new opportunities by taking advantage of integrating technologies like mobile, big data, IoT, and cloud services.

Consumerization has increased the self-service demands. This is also true for companies. A company needs to solve customer problems faster than the competitor if it wants to keep up with the business. This means that new services need to be available by an IT department as soon as possible; otherwise, it will be consumed from external sources. This has put an IT department into very difficult position. If the service cannot be acquired from the local IT department as a self-service manner, it can and will be purchased from the public cloud. Consuming services from the public cloud will lower the visibility of consumed services and their costs, and increase legal and/or security risks with possibility of data loss for the company [1].

The cloud, digitalization and consumerization have brought challenges too. Most of them have been related to security and privacy especially when consuming services from the public cloud. Companies, which consider that the security or privacy issues prevent them from using public clouds, will consider deploying their on-premise cloud infrastructure instead of using the public ones. Security and privacy are one of the reasons why a total

cloud adoption has not been seen on the market yet. In figure 1 Eurostat's latest research studies provide an example of cloud services adoption rates in the EU region. [2.]

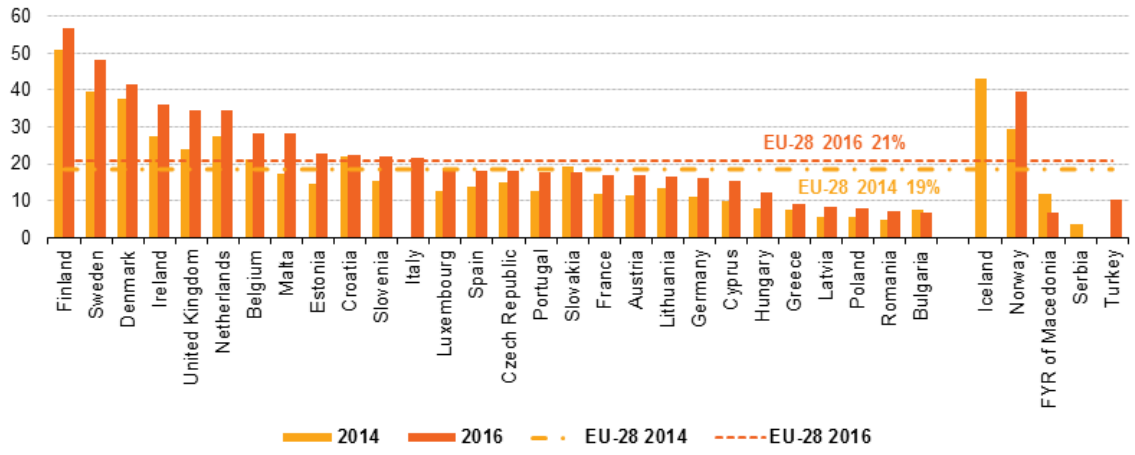


Figure 1. Eurostat statistics on enterprises' use of cloud computing services in the EU region. Reprinted from Eurostat [2].

Another factor that prevents companies from using cloud computing is the lack of knowledge. Eurostat and SUSE have found a similar indication in their research studies which show that one of the adoption barriers for cloud computing is the lack of skills. In order to understand cloud computing, many silos of technologies rather than a single technology or subject must be understood. [2;3.]

2.2 Where Does the Cloud Come from and in Which Form?

Cloud computing has been originated from the concept of hosting computer (CPU, memory, storage) resources on the cloud. Today's cloud computing has been evolved to consist almost anything related to IT (applications, virtual desktops, and such) that an organization would want to be hosted and offered through the cloud. In this context, it is relatively natural to start calling them cloud services instead of cloud computing. [1.]

The industry is usually using the "as a Service" ("aaS") term which is one way to segment the cloud technologies. The most commonly known "aaS" terms are an infrastructure as a service (IaaS), a platform as a service (PaaS) and software as a service (SaaS). These can be thought of as layers of a stack, each building on top of the previous one as shown in figure 2. [4.]

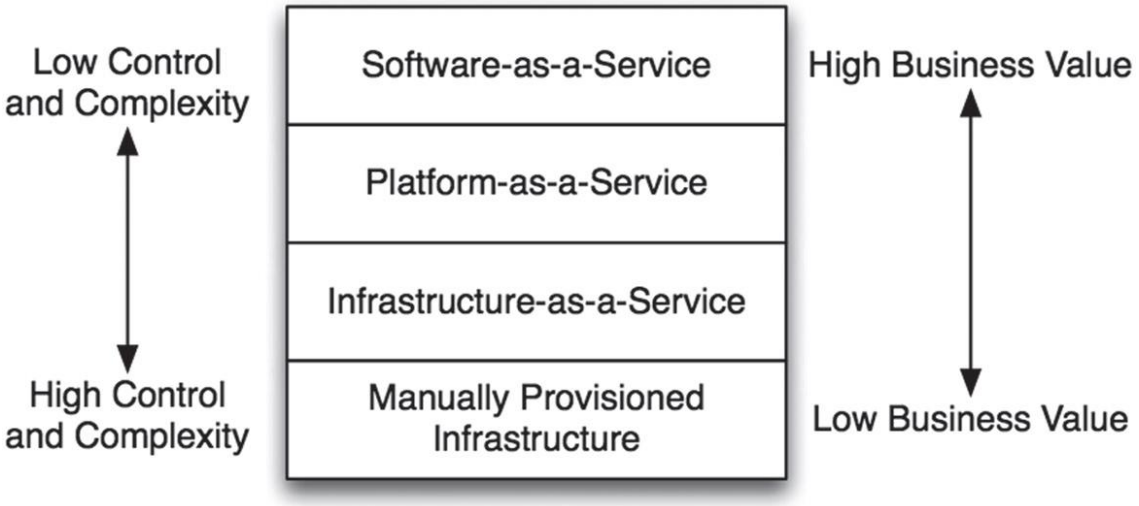


Figure 2. Different “aaS” with their example business values. Reprinted from OpenStack Cloud Application Development [4].

Another way to categorize the cloud technologies are by their deployment model. A private cloud is where the company has most of the control. When consuming services from the public cloud, the possibilities to control underlay hardware is near to zero. [4.]

How does the cloud infrastructure differentiate from the traditional (virtual data center) infrastructure? Both of them are heavily using virtualization. However, as virtualization is software (or technology), and as already described, the cloud can be referred to as all the services built on top of the virtualized infrastructure, the key is the delivery of the services whether they are data, software, or infrastructure. There is also another differentiator between them at the application level. The traditional infrastructure is mainly hosting a monolithic application on a single virtual machine, consolidating multiple virtual machines to a single host. The cloud instead is designed to host cloud-aware applications. These applications are designed to scale horizontally and are resilient for the shutdown of the virtual machines they are running. Figure 3 shows the difference between these two approaches. [5.]

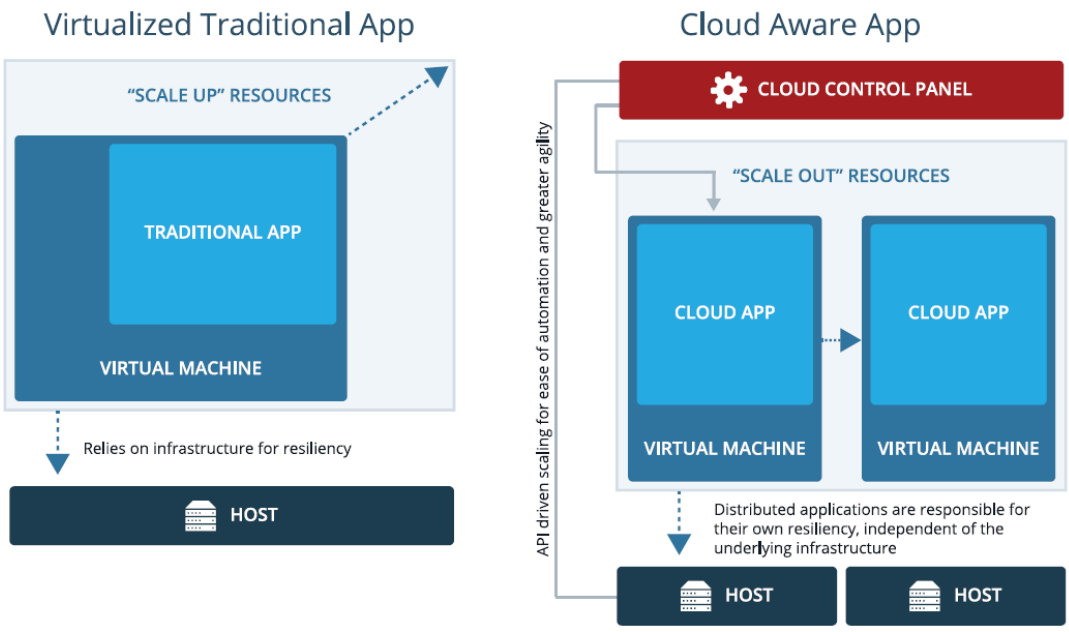


Figure 3. Difference between Cloud Aware App vs. Virtualized Traditional App. Reprinted from OpenStack Foundation [5].

The cloud has been evolved to provide containerized applications, serverless computing, big data analysis, and machine learning through the single pane of glass reducing application development work and speeding up the time it takes to develop an idea to a working product.

To conclude this topic, there are several benefits linked to consuming cloud services and also several different products and services to fulfil these needs. OpenStack has been named as a de facto standard platform for the private cloud market by the research company Forrester on their latest research studies [6].

3 OpenStack

3.1 Introduction to OpenStack

OpenStack describes itself as a cloud operating system, which helps to control a set of pools of computing, storage and networking resources. It provides a standardized API and a user interface to manage the hardware and the software resources it provides. [7.] This definition can be visualised as show in figure 4.

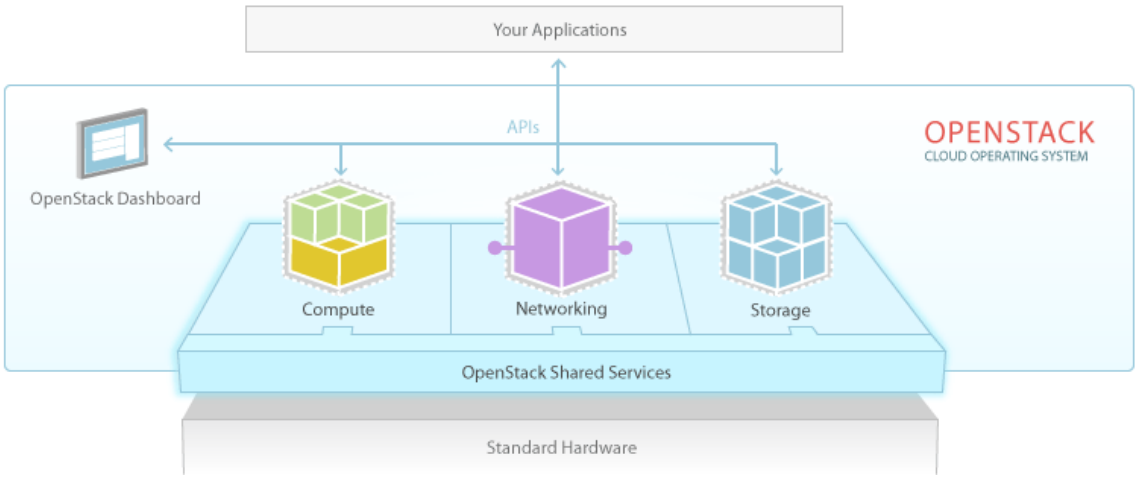


Figure 4. Diagram presenting OpenStack technology. Reprinted from OpenStack Foundation [7].

OpenStack can be seen to follow the National Institute of Standards and Technology's definition of the cloud computing. This definition is most widely referred to as declaration of describing "what cloud computing is" and what it "should" contain. OpenStack includes **on-demand self-serving** capabilities for the computing services it provides. These services can be accessed via a **broad network access**. OpenStack manages the **pools of resources** with a **rapid elasticity** and these resources and services can be **measured**. [8.]

The benefit of OpenStack is that it can be used to provide IaaS, PaaS or SaaS capabilities. It can be consumed as a part of a private cloud, a hybrid cloud, or used to provide public cloud services.

Lately the OpenStack has increased its popularity among the telecommunication (Telco) sector. The telecommunication sector has identified that OpenStack can help them with Network Function Virtualization (hereinafter *NFV*). OpenStack has been identified as one of the main components of the NFV, and it fulfils the functionality of the Virtualization Infrastructure Manager (VIM) in the reference implementation of the Open Platform for NFV (OPNFV). [9.]

Another sector that has found OpenStack useful is cloud service providers (CSPs). Many of them have started to move towards an open cloud solution. It is helping them to serve their customers better. Finnish ICT companies, like Datalounges and Nebula, are providing their cloud services from the OpenStack platform. [10;11.]

Also, other companies which are not diginatives like Netflix and Facebook, have found OpenStack to be ready for a production usage. Traditional manufacturing companies, such as BMW, have concluded that the industry standard on-premise IaaS solution can accelerate innovation and enhance operational efficiency through agility and automation [12].

One of the key differentiators to other cloud management platforms and orchestrators, like VMware vCloud Suite or Microsoft Azure Pack, is that the OpenStack is an open source solution and based on Apache 2.0 license. Due to this, OpenStack can be freely distributed and utilized. This can be counted as a benefit for the companies, which implement the OpenStack because they can avoid the vendor lock-in.

OpenStack can be chosen to be implemented from the source or as a distribution. Some companies offer OpenStack to be implemented as a service. Every method has its advantages and disadvantages. Table 1 includes decision markers based on business values, which can be used as a reference. [13.]

Table 1. Deployment model benefits by business values. Modified from OpenStack Foundation [13].

	"as a Service"	as a distribution	as a source
Time to production	Fastest	Moderate	Slowest
Configuration flexibility	Low	High	Highest
Software Customization	None	Depends on vendor policy	Unrestricted
Support	Vendor	Vendor	Community
External Costs	High	Med	Low
Internal Resource Requirements	Low	Med	Highest
Level of Resource Skill	Low	Med	High

OpenStack is modular by design, and most of the OpenStack functionalities are divided into different projects. From the launch of OpenStack, project counts have increased and consist today of over 19 different projects. To provide a structure around these diverged set of projects, the OpenStack Foundation has setup structured project designations. These designations are core, incubated, library, gating, supporting, and related. Often, when OpenStack is mentioned, the core designation is meant.

Another way to simplify the OpenStack structure is that the projects are divided into the core and complimentary services. The core services are considered as mandatory to get the OpenStack implemented and running and the complimentary services provide additional functionality and services around it. [7;14,6-19,85.] These will be described in the next two chapters.

Even though OpenStack is said to be an operation system, it needs to be deployed on top of the Linux operation system, and additional components are needed, like a data base and a messaging layer. These are not included in any of its own project lists. The OpenStack does not provide its own virtualization hypervisor but consumes and supports multiple hypervisors like KVM, XEN, Microsoft Hyper-V or VMware ESX through the vSphere.

OpenStack is developed and released every six months. These releases are coordinated in the OpenStack Summits. The OpenStack release is supported by nine to fifteen months after its initial release. Every release is numbered and codenamed. At the time of writing this thesis, the 15th release of the OpenStack platform was announced with a codename “Ocata” and numbered as 2017.1. [15;16;17.]

For most companies, keeping up with the OpenStack release cycle is usually hard to follow, as stated by the Forrester research. For this reason, the OpenStack distribution vendors are supporting an OpenStack release for a longer period. [18.]

3.2 OpenStack Core Services

OpenStack includes a governance model, which every project needs to pass before it is listed as an OpenStack core service [14]. As this thesis concentrates on how to design a cloud platform with customer specific set of services, the following components in table 2 are considered to be core services of the OpenStack regardless of the fact that all projects in table 2 have not passed the OpenStack governance model.

Table 2. Core OpenStack services considered as context of this thesis. Data gathered from SUSE [19].

OPENSTACK SERVICE	PROJECT NAME	DESCRIPTION
Dashboard	Horizon	Provides a web-based, self-service portal used to interact with underlying OpenStack services, such as launching instances, assigning IP addresses, and configuring access controls.
Compute	Nova	Manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling, and decommissioning of virtual machines on demand.
Networking	Neutron	Enables network-connectivity-as-a-service for other OpenStack services, such as OpenStack compute. Provides an API that enables users to define networks and the attachments that go into them. Has a pluggable architecture that supports many popular networking vendors and technologies.
Object storage	Swift	Stores and retrieves arbitrary, unstructured data objects via a RESTful, HTTP-based API. It is highly fault-tolerant with its data replication and scale-out architecture. Its implementation is not like a file server with mountable directories. Instead, it writes objects and files to multiple drives, ensuring that data is replicated across a server cluster.
Block storage	Cinder	Provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.
Identity	Keystone	Provides an authentication and authorization service for other OpenStack services. Provides a catalog of endpoints for all OpenStack services.
Image service	Glance	Stores and retrieves virtual machine disk images. OpenStack compute makes use of this image service during instance provisioning.
Telemetry	Ceilometer	Monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.
Orchestration	Heat	Orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.
Application programming interface (API)	OpenStack API	Provides application programmng interfaces for block storage, compute, identity, image services, networking and other OpenStack components.

Without all the services listed in table 2, it would be possible to create a cloud environment. However, necessary cloud characters, like measurability, could not be met.

3.3 OpenStack Optional Services

OpenStack includes 13 optional services that enhance and expand the usability of OpenStack technology. Some of them are more mature than the others and under active development. Before using them in production, supportability and the roadmap should be reviewed.

From the optional services list, services listed in table 3 are considered to be useful in the context of this thesis.

Table 3. Supplementary OpenStack services considered to mention of context this thesis. Data gathered from Sarat and Shrivastwa [20].

OPENSTACK SERVICE	PROJECT NAME	DESCRIPTION
Sahara	Elastic Map Reduce	Sahara service is the Big Data service of OpenStack; it is used to provision a Hadoop cluster by passing a few parameters.
Trove	Database	Trove is the Database as a Service component of OpenStack.
Ironic	Bare-Metal Provisioning	The Ironic service allows bare metal provisioning using technologies such as the PXE boot and the Intelligent Platform Management Interface (IPMI).
Designate	DNS Service	The Designate service offers DNS services equivalent to Route 53 of the AWS.
Magnum	Containers	Magnum introduces Linux Containers such as Dockers and Kubernetes (by Google) to improve migration option.
Murano	Application Catalog	Murano is an application catalog, enabling application developers and cloud administrators to publish various cloud-ready applications in a catalog format.

These services provide additional capabilities for the end-users of Metropolia Education Cloud Platform. Especially Murano is important as it provides an application catalog that makes cloud-ready applications available.

3.4 OpenStack Architecture

Because OpenStack is modular and highly configurable, with different back ends and network configuration options, there is no single way to present it. One example to visualize the OpenStack architecture and the communication path between different components is shown in figure 4.

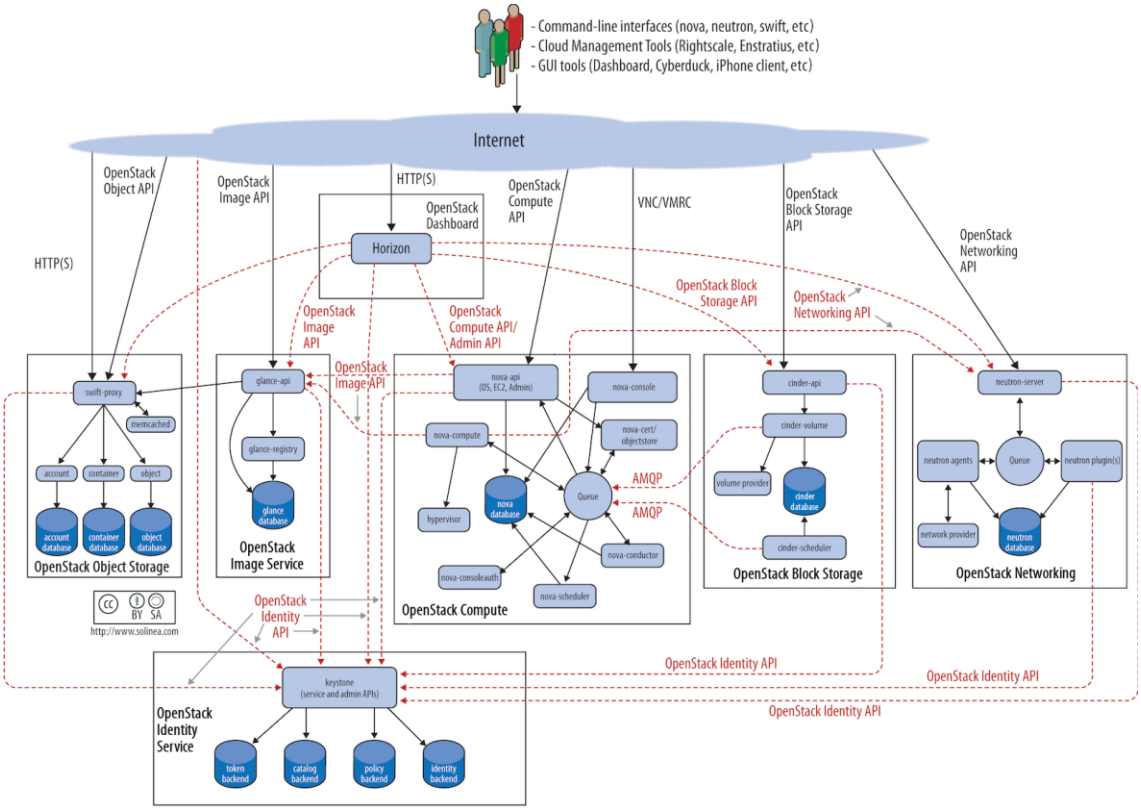


Figure 4. The OpenStack logical architecture. Copied from OpenStack Foundation [21].

From the architectural point of view, OpenStack services (projects) are loosely coupled, and communication between them is mostly done via Application Program Interface (API). This architecture makes OpenStack very scalable. As for reference, CERN had already over 5,500 Compute node hosts in their OpenStack environment in February 2016 [22].

OpenStack services are commonly divided into a host (a node) that is a physical machine. The nodes are usually named by the functionality they provide. Controller is a node that usually serves the management services, whereas compute node provides computing capacity. Other possible node types are storage and network, as shown in figure 5.

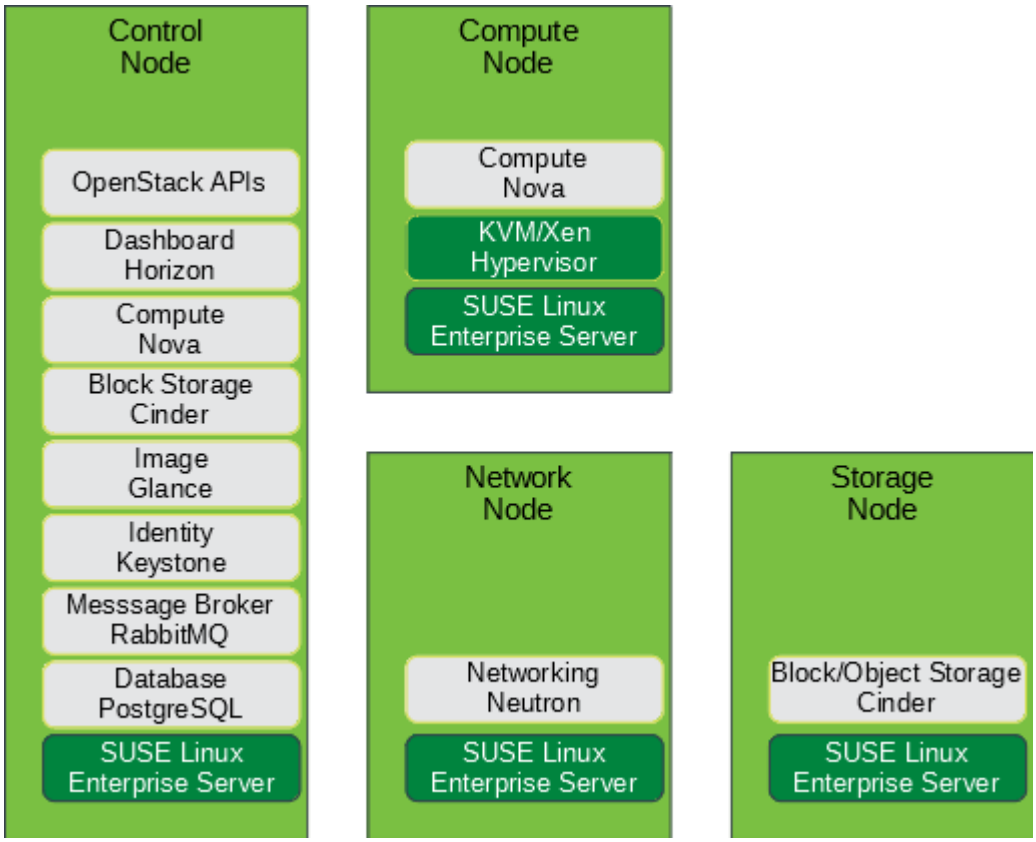


Figure 5. Simplified picture showing an example the OpenStack services setup on top of each node type. Data gathered from SUSE [19].

Every service can be deployed to a dedicated host node. However, this increases the risk that the whole service is unavailable if any of the host serving shared services becomes unavailable. This risk is Usually mitigated by high availability of services, which is an issue that will be discussed in the next chapter.

3.5 Highly Available OpenStack Services

Because OpenStack is used for managing several end-user workloads, any disruption to OpenStack services will usually affect all of them. To avoid system downtime and preventing data loss that might occur during the interruption, high availability is used to mitigate the risk. High availability services are usually implemented with redundant hardware and avoiding any single point of failure (SPOF). In the event of failure, affected services are taken over by another system. This kind of a setup is usually described as an active/passive configuration. If service is running on all nodes in the cluster, and a load can be shared between the nodes, the setup is called an active/active configuration. [23.]

OpenStack components, which should be made as highly available, are the services running on the controller and the network node. These are providing shared services, and an interruption of any of these components affects the whole cloud environment. A minimum viable cluster size for high availability is two host nodes. In this case, there is a possibility to run into a situation called “split-brain”. In this situation both members of the cluster think they are the only one that survived a catastrophic situation and will try to fence the other node to ensure it does not start and run the cluster controlled services. As this cannot be done, the situation will lead to a data corruption. To avoid split-brain situations, a quorum with an odd number of cluster members is used. A commonly used technology for the high availability is either Pacemaker or Keepalived. Usually the distribution vendor of the OpenStack has chosen either one to be used in their OpenStack implementation. SUSE has chosen a Pacemaker and this architecture will be covered in Chapter 4.7 Highly available SUSE OpenStack Cloud. [23.]

Initially the OpenStack was designed to run cloud native applications, and thus, there is not a general concept for high available compute node services. Distribution vendors have solved this in their own way. Currently, there are three ways to accomplish compute node high availability. SUSE and Red Hat are using OCF agents to protect compute node failure. A compute node can be added as a remote node to the Pacemaker cluster. In the case of a host failure, a virtual machine run on the failed host is started automatically on a different host. The other two ways are not relevant for the content of this thesis. The OpenStack community has started to design an official common solution too. [24.]

High availability of the storage backend (Cinder) is implemented by choosing a technology that supports it natively. Ceph is usually considered to be highly available by design. [25.]

4 SUSE OpenStack Cloud

4.1 Introduction to SUSE OpenStack Cloud

SUSE OpenStack Cloud is a distribution of OpenStack. With SUSE OpenStack Cloud, an enterprise can easily manage its own private cloud infrastructure. SUSE OpenStack Cloud is designed to be fast, secure, and easy to deploy.

SUSE OpenStack Cloud 6 is the sixth release and based on the Liberty (OpenStack 2015.2) release. SUSE OpenStack Cloud 6 uses SUSE Linux Enterprise Server as an operating system, the OpenStack as a cloud orchestrator software and Crowbar and Chef as a deployment tool. It has a broad hypervisor support. [25;26.]

SUSE OpenStack Cloud brings available services that have been tested and proven to be enterprise-ready. This can be counted as a benefit when implementing a distribution version of the OpenStack. Some of the latest functionalities of OpenStack are provided as a technology preview status. This means that they are included, but not supported. An example of this is the OpenStack Bare Metal (Ironic). [26.]

SUSE OpenStack Cloud is sold as a subscription based model. Options are one- or three-year priority support per a purchased node type. The initial SUSE OpenStack Cloud Control Node and SUSE OpenStack Cloud Administration Server have been bundled together. Additional SUSE OpenStack Cloud Control Nodes, Compute Nodes and Swift Storage Nodes are sold separately.

From the architectural point of view, SUSE OpenStack Cloud is deployed to four different node types:

- one Administration Server (Admin Node)
- one or more Control Nodes
- several Compute Nodes
- none or several Storage Nodes

Figure 6 presents a layout of these nodes and how they communicate with each other.

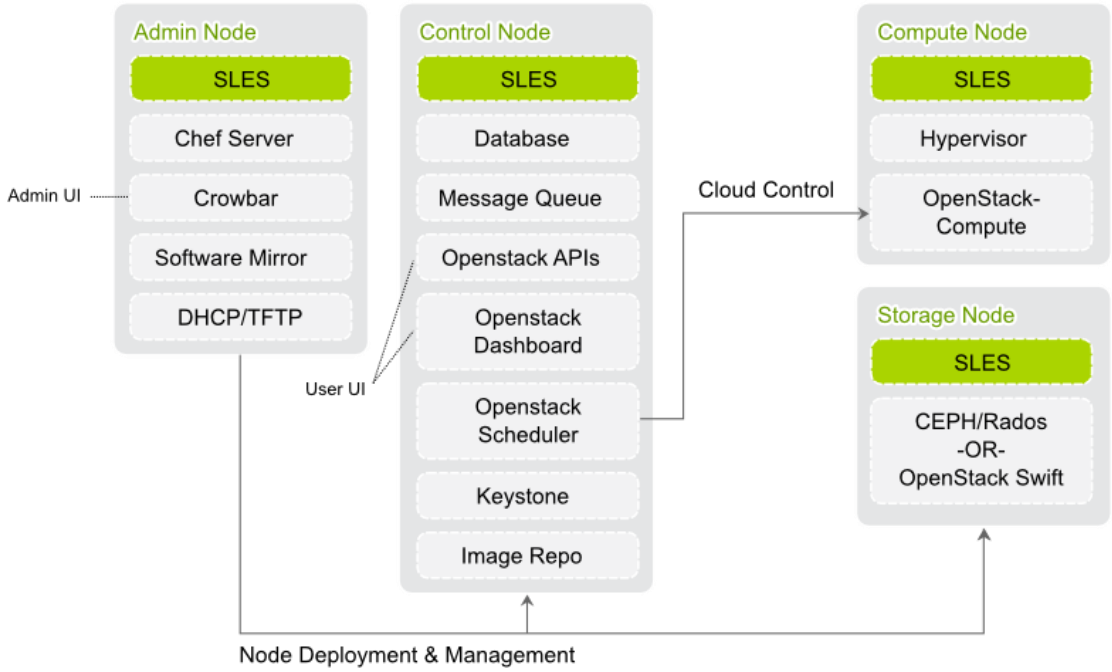


Figure 6. SUSE OpenStack Cloud architecture. Copied from SUSE [25].

Every node type has its own functionality. They serve in the SUSE OpenStack Cloud infrastructure.

Administration Server is a centralized deployment tool for SUSE OpenStack Cloud. It contains all necessary services and repositories needed to deploy and manage all other nodes in SUSE OpenStack Cloud. It provides core infrastructure services: DHCP, DNS, NTP, PXE, and TFTP, along with Crowbar and Chef. Crowbar is used as a provisioning tool for the node deployment. Chef is used for the installation of the OpenStack components and for automating the configuration management across SUSE OpenStack Cloud. The figure 7 shows Administration Server services and how they interact with Node in SUSE OpenStack Cloud. [25.]

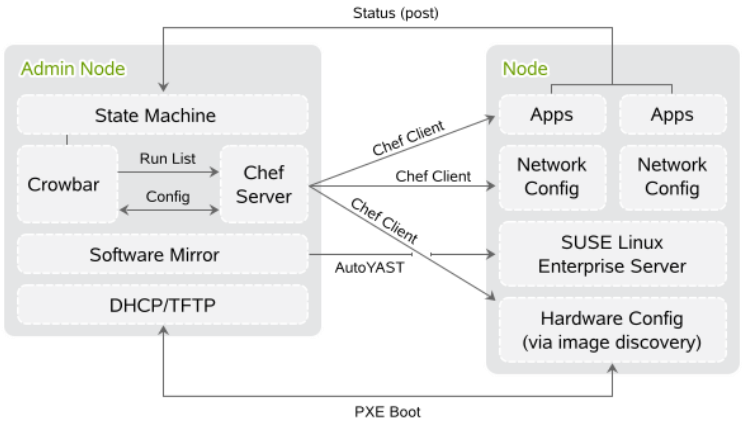


Figure 7. Administration Server services and communication path between Administration Server and Node in SUSE OpenStack Cloud. Copied from SUSE [25].

Administration Server maintains a set of configuration recipes of Chef, which are called barclamps. With help of these recipes the cloud deployment can be mostly automated. Barclamps can be divided between Crowbar (infrastructure itself) and OpenStack. Crowbar barclamps are designed to setup and configure all the nodes. OpenStack barclamps are dedicated for the OpenStack service configuration. Barclamp is created from a template. Once barclamp has been created, it can be edited and parameterized to correspond to the current setup. The Chef server is handling the barclamp configuration delivery. OpenStack services are usually deployed in the order founded in the Crowbar web interface. A deviation from the order might be needed if any of the components is not used, like OpenStack Swift. [25.]

One control node can host all the shared services needed to orchestrate SUSE OpenStack Cloud. This is suitable for a proof of concept environment and not recommended for a production usage. Usually these services are divided among several control nodes. A preferred option is to setup a control node cluster for high availability. This issue will be covered in a later chapter. A control node can also host network functions (Neutron). In a wider environment, and/or for security reasons, they can be divided into their own node types. [25.]

Compute nodes are the pool of servers where the workloads are running. They are running nova-compute service, and the service manages the deployment, and the start and the stop of the workloads. SUSE OpenStack Compute node supports multiple hypervisors, such as Hyper-V, KVM, VMware vSphere, Xen, Docker, and z/VM, but only one hypervisor can be running at the time. [25.]

Depending on the type of the storage that is chosen to be used in the cloud environment, the storage nodes are setup as a pool of machines that provide an object or a block storage to cloud workloads. The storage nodes are chosen as a default designation to provide OpenStack Swift, Ceph or Cinder's raw local disk storage. [25.]

4.2 SUSE OpenStack Cloud Network

SUSE OpenStack Cloud requires a complex network setup consisting of several networks that are configured during the installation phase. With a default installation of SUSE OpenStack Cloud five logical network segmentations are used. They are the admin network, the storage network, the private network, the public network, and the software defined network (SDN). These networks are presented in figure 8. The admin, the storage, the private, and the SDN networks are considered to be exclusive to the inner-cloud communication, and the public is used to provide external inbound and outbound (North-South) communication. For security reasons, it is advised to isolate the VM network traffic. [25.]

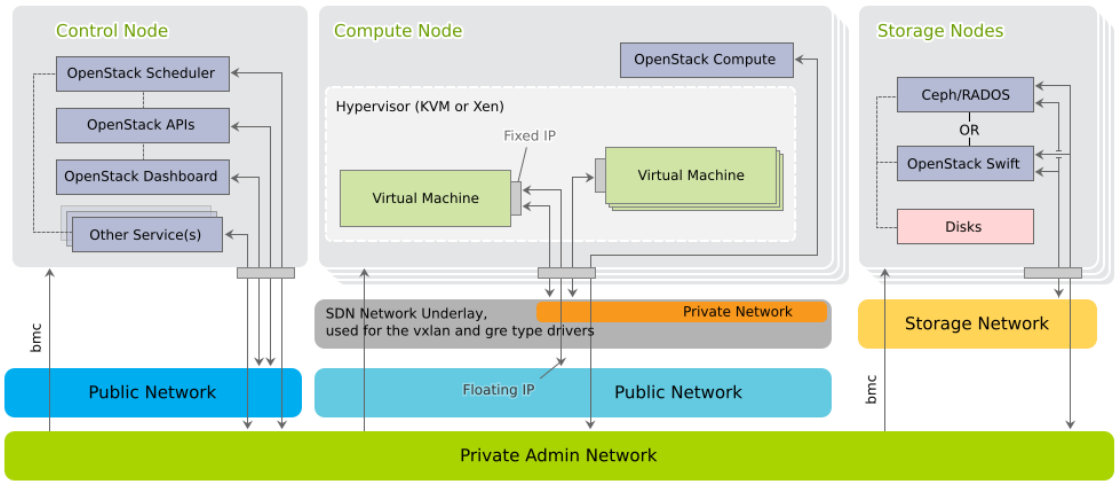


Figure 8. Default network layout of the SUSE OpenStack Cloud. Copied from SUSE [25].

Every segment is using a class C network range by default. This limits the maximum number of systems used in SUSE OpenStack Cloud. SUSE is providing the YaST tool to make the changes to the address scheme easier. With the same tool, changes can be made to the network mode used. A teaming mode is advised to be used since it does not introduce SPOF to the network operations. Teaming is also required to be used on

an HA setup. This increases the required network cards count to two. An example a teaming setup is presented in figure 9. [25.]

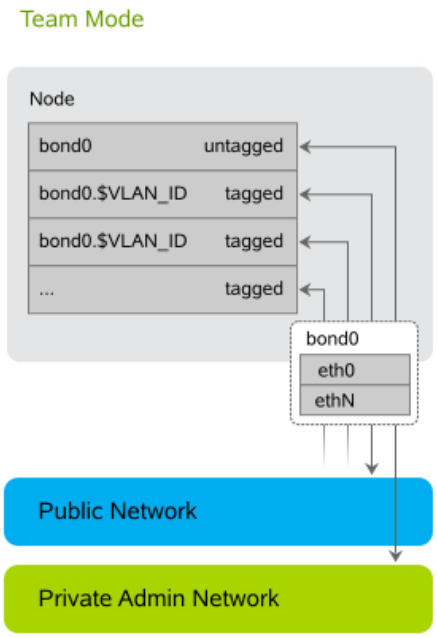


Figure 9. Example of teaming setup with two network cards. Copied from SUSE [25].

More complex network setups require manually editing the network barclamp template, which is located in */etc/crowbar/network.json*. An example of a complex network setup would be the use of more than one teaming inside SUSE OpenStack Cloud.

4.3 SUSE OpenStack Cloud Storage

SUSE OpenStack Cloud provides two types of storage to be consumed on the cloud. An ephemeral and a persistent storage. The ephemeral storage exists only as long as a VM exists. The persistent storage remains after the termination of workload. Both have they preferred use cases. The ephemeral disk should be used, when a virtual machine is considered to be disposable. This is very a cloud native way to consume workloads from the cloud. The persistent storage can be used to store a virtual machine or the data shared with them. [25.]

4.4 Highly Available SUSE OpenStack Cloud

As already stated in chapter 3.5, high availability can mitigate the risk of the system downtime. SUSE OpenStack Cloud can be made highly available so in the case of a failure event, it continues to provide the services by migrating or starting services on another control node in the cluster. [25.]

Only the DNS and the NTP service from the administration server are considered to be crucial for the OpenStack services to run properly. These services should be run on multiple nodes simultaneously. Otherwise the administration server does not take part in providing any of the services towards the end users, and therefore only the disaster recovery is advised to be planned upfront. [25.]

Instead of assigning OpenStack services to an individual control node, it is advised to create one or several dedicated clusters to run these services. A common approach is to start with one cluster. Another cluster for network services might come into question when a network traffic exceeds the capacity of the service cluster. Data services, including a database and a message queue, can be separate to another cluster. Dividing the services to multiple different clusters increases the required host count and operational work. [25.]

A typical one cluster setup is represented in figure 10. The OpenStack services have been divided between three control nodes in an active/active way. The database and the message queue are components that are setup as an active/passive way. [25.]

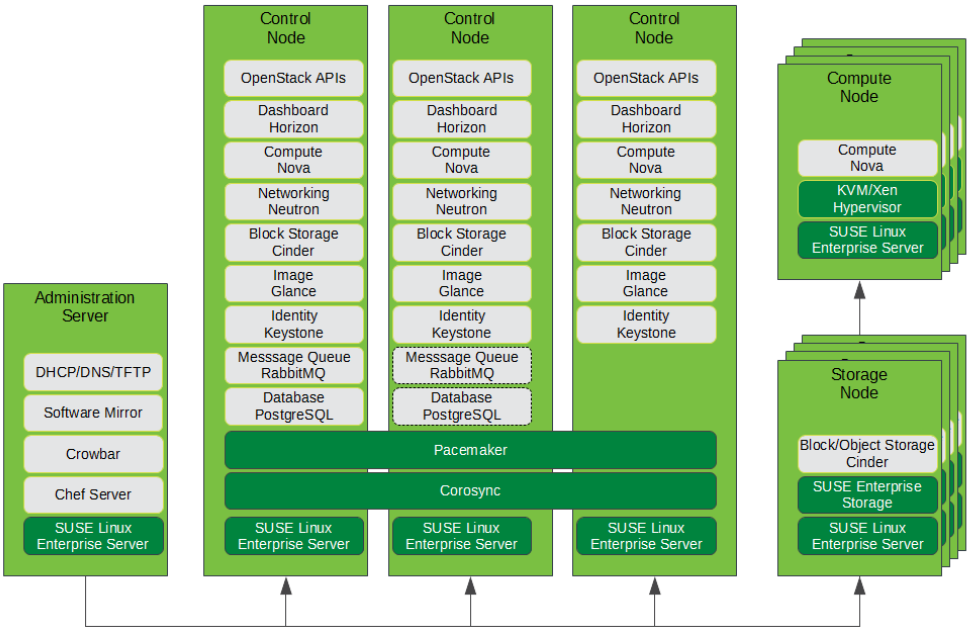


Figure 10. Typical SUSE OpenStack cluster setup. Data gathered from SUSE [19].

The Pacemaker component is used as a cluster resource manager. The Pacemaker's task is to manage the start, the stop, and the recovery of the OpenStack services in an automated way. [25.]

The Pacemaker installation and configuration is automated with the Pacemaker barclamp. After the setup of the control cluster, it can be used as a designation of the OpenStack services deployment, so that the services are made highly available. [25.]

Compute nodes are made high available via the Pacemaker-remote service. This minimizes the downtime suffered by a failing compute node. The individual virtual machines cannot be made highly available with this solution. [25.]

A storage is considered to be highly available when using OpenStack Swift or Ceph as a storage solution. [25.]

5 Principles of Design for OpenStack Cloud

5.1 General Process

Historically the OpenStack environment has been challenging to design. This is mainly due to the fact that the OpenStack environment is installed in parallel to the existing infrastructure and hardware and the needs of the future while still maintaining scalability and flexibility.

A general approach used for designing the OpenStack cloud architecture has been by creating a business or a technical use case. Once the use case has been validated the requirements are gathered. The requirements can be related to functionality, security or legal (regulatory compliance) aspects. The most common starting point for the OpenStack cloud architecture is the general-purpose architecture. This can address up to 80 percent of the potential use cases available.

The general-purpose architecture supports common operation systems and applications found in the companies, and this is one of the reasons why private cloud should be built with it. If the usage of the cloud is not clear yet, it is also advised to start with the general-purpose architecture. It is balancing computing, networking and storage resources. The other seven types of the pre-design OpenStack architectures are focusing on compute, storage, network, multi-site, hybrid, or massively scalable architecture. [27.]

As the general-purpose architecture fulfils the requirements of the Metropolia Education Cloud Platform, it is used as a starting point for final year project and the next chapters.

5.2 Compute Resource Design

Many different points need to be considered when designing OpenStack computing resources. A good starting point is to choose how many pools are needed to provide computing resources. Resource pools are usually segregated by an availability zone or by host aggregate rules. This ensures that the end users are able to distribute their workloads across different pools so that the high availability is achieved in the event of a pool failure. This will also ensure the quality of service (QoS) requirement as in this

way the resources can be limited and a noisy neighbour issue, where a tenant's workload can affect another tenant's workload, can be avoided. [27;28.]

It is advisable to choose hardware that supports virtualization natively. In the CPU, this is ensured through hardware assisted virtualization capabilities and functionality of the extended page tables (EPT). [29.]

Overcommitting computing resources, like a memory and a CPU, is a commonly used method in the cloud environments. This is due to the fact that the workloads do not consume the resources all the time. However, it is recommended to review the hardware vendor recommendations for overcommitting the computing resources and monitor their consumption. If this information is not available, it should be gathered with the help of the proof of concept environment. [27;30.] Also different technologies supported by the hypervisor could be considered. For example on KVM, the Kernel Samepage Merging (KSM) de-duplicates and consolidates private memory pages and can as such improve virtual machine density.

A starting point for the initial deployment is the amount of workloads. This gives the minimal viable environment that should be targeted. Once the amount and the average size of the workloads are known, this information can be mirrored to OpenStack flavours. Flavours define the size of a virtual server on OpenStack. If needed, customized flavours can be created. [31.]

To forecast the cloud consumption is not an easy task to do. There are several third-party monitoring and forecasting tools available. At minimum, a standard open source monitoring tool like Nagios, Icinga or Zabbix is advised to be used to track consumption information of the CPU, memory, disk and network. [30.]

It is also advised to plan how to add computing capacity to the pool when demand requires it. [27.]

5.3 Cloud Network Design

Designing and choosing a network architecture is the most crucial part of the cloud computing environment. If the network is not planned carefully enough, there is a high risk of an unauthorized access and potential data loss. Design affects the overall

performance and the throughput of the network. It is advisable to isolate the network segments by a physical or a logical separation. At minimum, there should be three logical segments; 1) public, 2) administration, and 3) internal cloud communication. [27;30.]

A network connectivity for cloud workloads is provided by the OpenStack network service. There are initially two ways to accomplish it: A Nova-network (legacy networking) but as this does not support self-service capabilities, it will not be dealt with in this thesis. Instead, OpenStack Neutron is the preferred component as it supports self-service capabilities. The Neutron supports the segmentation of the project networking. The project networks are considered to be self-service networks. They are encapsulated with the generic routing encapsulation (GRE), or with virtual extensible LAN (VXLAN), or VLAN tags. Choosing an encapsulating method usually depends on the used hardware and capabilities it has been licensed to support. When using the Modular Layer 2 (ML2) plugin with Neutron, multiple network types and mechanism drivers can be used simultaneously. [27;30.]

5.4 Design a Cloud Storage

As the storage can be found from several places in the OpenStack cloud, at the designing phase, storage requirements should be reviewed carefully.

With the ephemeral storage, the OpenStack supports three different persistent storage types, which can be provided as a service: an object storage, a block storage, and a file system storage. The object storage is implemented as the OpenStack Swift. The block storage is implemented as the OpenStack Cinder project. The file system storage is implemented as the OpenStack Manila project. [27;30.]

The general-purpose architecture balances between the capacity, the costs, and the performance. Optimally the lowest cost per a terabyte, which can manage to achieve the performance and the capacity expectations, is usually chosen for the storage. [27.]

Ceph is usually recommended as a cloud storage backend because it is a scalable and distributed. It suits for the OpenStack storage backend well. It can be used for the Cinder and Glance project backend. Ceph is the most used Cinder driver based on the latest OpenStack user survey [32].

5.5 Designing a Secure OpenStack Cloud

Security has been one of the key aspects for cloud consumers. OpenStack is built to be secure. However, OpenStack security needs to be maintained and monitored continuously.

As the security of the OpenStack is a broad topic, only general principles to design a secure OpenStack Cloud that are also related to Metropolia Education Cloud Platform are reviewed in this chapter.

For network security, the network should be divided into three logical or physical segmentations at the minimum. Unnecessary access to these segmentations of the network should be limited. Limitation can be done with firewalls.

Once the network security has been taken care of, the Secure Sockets Layer (SSL) with the Transport Security Layer (TSL) should be implemented for OpenStack services. This ensures that the communication between the cloud services and the consumers has been encrypted, and the integrity and confidentiality of the data is protected. [30.]

Security information (including logs and event information) that OpenStack generates should be gathered into a dedicated logging server for future monitoring and analyzing purposes. An example of an open source stack for logging is ELK (Elasticsearch, Logstash and Kibana). [30.]

To lower security risks at the operation systems level and software level, a common practice is to setup an update process to keep the currency up-to-date. This helps to mitigate one aspect of the security risks. [30;33.]

However, as the use of Metropolia Cloud platform is so specific, there are areas where the security requirements have been decreased. An example of this is the nested virtualization capability, which has been enabled at the hypervisor level.

To summarize, once OpenStack has been secured, there are several places outside OpenStack where additional hardening should be done. One example of this is network switches.

6 Metropolia Education Cloud Platform

6.1 The Concept

The concept of Metropolia Education Cloud Platform was invented by Harri Ahola, a Senior Lecturer at the Helsinki Metropolia University of Applied Sciences, Finland. To address increasing demand of providing highly valued educational services and to provide diverse methods of e-learning, Ahola recognized that the only solution was to deploy Metropolia's own private cloud infrastructure.

The business use case for Metropolia's Education Cloud Platform can be easily calculated. Even if the public clouds are able to provide workloads based on the consumption based cost model, the overall cost of keeping a course in the public cloud immediately rises over the budget cost levels.

6.2 Technical Requirements for the Metropolia Education Cloud Platform

Most of the technical requirements of the Metropolia Education Cloud Platform are related to supporting information technology studies at the Communications and Network Engineering Department (especially IoT ["Virtualization"] and Cloud Computing).

The main functionality of the Metropolia Education Cloud Platform is to provide and support IaaS capabilities. The OpenStack is suitable for providing that, and on the top of it, a service offering that supports end-users to build and consume PaaS and SaaS services can be easily built.

Because Metropolia provides the Education Cloud platform for research and thesis work, it should not be tied to support only a few technology vendors. The openness and supportability for multiple different technologies are required as well. The OpenStack fulfils this requirement perfectly. It has several technology vendors as a foundation member and has a long list of technologies supported.

One of the reasons why public clouds are not always suitable for the courses taught at the university of applied sciences is that they are a highly restricting. They do not support all the technologies and functionalities required. One of the functionality, which is often

needed is nested virtualization. This enables running a virtualization on top of a virtualization. It is an easy way to test and learn new technologies, as there is no need to purchase dedicated hardware to it.

At Metropolia, there is a common will to increase the use of self-service tools and automation levels. This translates to service offering, where the teacher could setup his/her own learning environment for the course he/she teaches by himself/herself from the self-service portal.

The cloud needs to be highly available. As today's studies are not restricted to classroom education and as students might want to study even on Saturday evenings, the learning environment needs to be fully working when needed.

Metropolia has extensive hardware resources from different vendors, and these all should be easily integrated into and used in the Metropolia Education Cloud Platform.

Discussions with Ahola focused on the need of supporting the computing resources and services. He also suggested that the Education Cloud platform could provide other "as a service" capabilities, like Big Data (Hadoop) as a Service and Content Delivery Network (CDN) for video streaming.

6.3 Proposed Reference Architecture

After analyzing the use case and the technical requirements, and validating that they can be met with the selected OpenStack distribution, a reference architecture was designed.

This phase included multiple designing sessions where requirements and architecture design were gone through again.

As Metropolia had already made a decision on a hardware vendor and purchased the hardware (Cisco Unified Computing System (UCS) blade server devices, built by Cisco Systems, Inc.) and the storage solution (EMC) to use within the Metropolia Education Cloud Platform, the starting point for the technical architecture was to use SUSE's reference architecture for SUSE Cloud integration with Cisco UCS. [34.]

With Cisco hardware, there is a possibility to leverage integration between the SUSE OpenStack Cloud and Cisco UCS Manager as shown in figure 11. This integration enables doing basic maintenance tasks, like rebooting the server, from the SUSE Cloud Administration server.

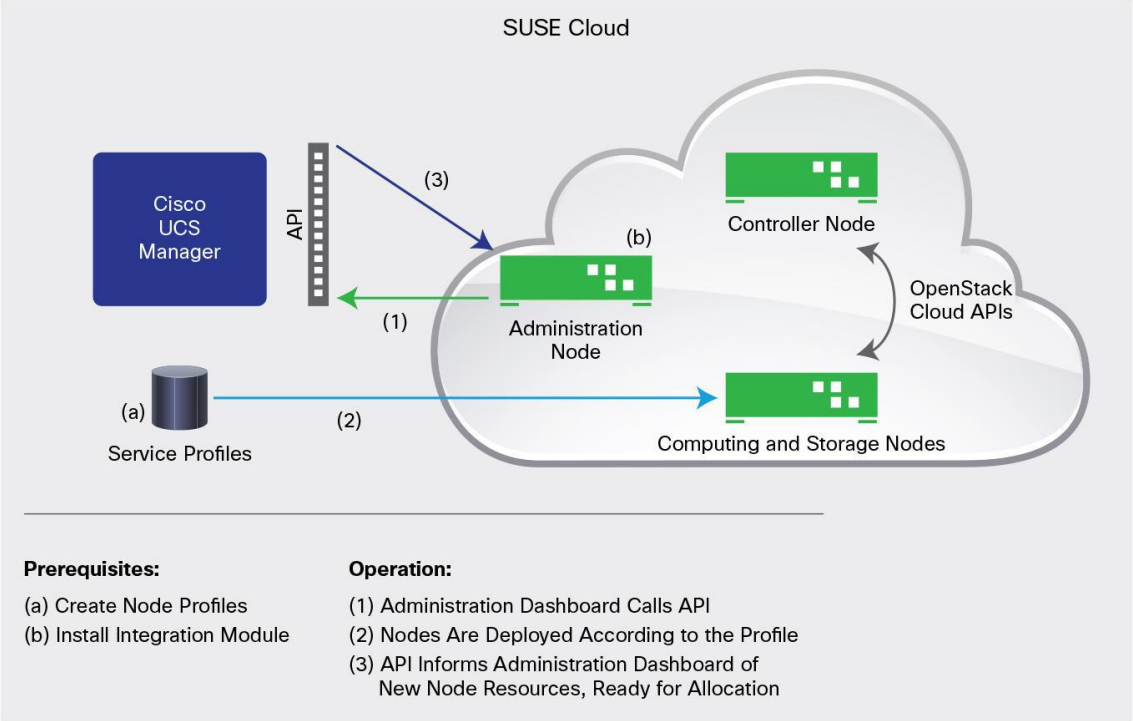


Figure 11. SUSE OpenStack Cloud and Cisco UCS Manager integration communication path. Copied from Cisco [34].

The main functionality of the Metropolia Education Cloud Platform is to perform an e-learning environment for Metropolia faculty and multiple students simultaneously. For this particular reason, there is a predefined availability requirement. This requirement could be met by using a highly available cluster for the OpenStack shared services. To keep operational and maintenance costs low, only one service cluster was designed to be used. This was also justified, as there was no secondary server room, a data center, or a disaster recovery site available. If one of these becomes available, the architecture needs to be reviewed. As the service cluster was located inside the server room, a split-brain situation was considered to happen unlikely. The service cluster node count was lowered to two hosts for this reason.

For a cluster STONITH device a SBD (STONITH Block Device) was chosen to be used. SBD is a storage based fencing mechanism. This requires a shared storage to be visible on both controller nodes. It is a small <100 MB device.

For a database (PostgreSQL) and a message queue (RabbitMQ), a DRBD was chosen to be used. This replicates the data available to another node to be consumed if/when the active host becomes unavailable. For this usage, a dedicated 50 GB disk should be available on both hosts.

To meet the architecture design and estimated resource requirements, the already purchased Cisco hardware was advised by Olli Tuominen to be refurnished with a correct amount of memory and disks. This technical information is given in Appendix 1.

The principles of avoiding a SPOF were used in the network design. Every node contained four network interfaces. Two separate teaming interfaces were created on top of them. A teaming mode 4 (802.3ad) was chosen to aggregate the interfaces to the group. The first teaming group was used for management and internal cloud communication. The second was used as a public network communication. This helped to secure the network and avoid an unauthorized access to running workloads.

For the tenant network, VXLAN was chosen to be used as the OpenSwitch driver (a Neutron component). There are several benefits when using a tunnelling protocol with the OpenSwitch driver, which is why the VXLAN was introduced to be used inside the Metropolia Education Cloud Platform. It gives more flexibility than the traditional VLAN and has been proven to be more robust than the GRE protocol. VXLAN has been a preferred overlay encapsulation method since Icehouse release. However, if different hypervisors will be introduced to the Metropolia Cloud platform, this needs to be re-evaluated as all tunnelling protocols are not equally supported by different hypervisors.

Due the security constraint, a bastion network configuration was chosen to avoid an unauthorized access to the administration network. Otherwise the network was segmented with five logical segments.

An IP address range was chosen so that the tenant networks will have enough free IP addresses to be consumed for the workloads and that the infrastructure will have room

to expand if needed. As this information can be misused, it is not presented with in this thesis. Default network ranges can be seen in Appendix 2.

A domain name needs to be available for the usage of the cloud. A sub-domain from Metropolia Labnet was reserved and used for this purpose.

To avoid routing all the network traffic through the controller nodes, the Distributed Virtual Router (DVR) setup was chosen to be used in the Metropolia Education Cloud Platform. This also reduces the risk that the controller nodes come to a bottleneck for the network traffic in and out of the cloud environment.

There are several network tuning options within SUSE OpenStack Cloud. These should be matched with the underlying network and hardware capabilities. As for Metropolia Education Cloud Platform, the MTU value 9000 should be chosen for the storage network. This will increase the storage network throughput drastically.

The overall goal was to use the already purchased EMC storage solution for the Metropolia Education Cloud Platform as a primary storage solution. However, it was found to be a problematic as the EMC VNXe 3200 storage solution has not been certified and supported by the OpenStack natively. It can be used as a Cinder volume disk, presented to a storage node as a local disk. When consumed this way, it can be considered as an SPOF. The best storage option for the SUSE OpenStack Cloud is Ceph. Even though the Ceph is a relatively new technology, it has been proven to be scalable, reliable, and easy to use with the SUSE OpenStack Cloud. Metropolia was advised by Tuominen to procure a suitable server hardware to be used with Ceph. This increased the server node count by four nodes. It is the minimal configuration of the Ceph that provides enough capacity and performance, and it is still supported by SUSE. The storage for Cinder and Glance can be switched to Ceph, as it becomes available. In the meantime, alternative storage design was chosen to be used, where the storage device is presented to a storage node as a local disk.

For the virtualization, a kernel-based virtual machine (KVM) was chosen to be used as a primary virtualization technology. It is the most commonly used hypervisor of the OpenStack environments, and good references and experiences can be found in other SUSE OpenStack Cloud implementations.

Once all these variables have been considered and taken into account, the actual implementation of the design can be started. Usually, problems can be found that must be solved before the actual production use.

7 Conclusion

OpenStack is the technology that will be used for the next generation's software-defined data centers. It is essential to understand how OpenStack works and how it can be made highly available. This can help to understand public clouds, too. This goal was accomplished in the project documented in this thesis.

Designing information technology architecture is not an easy task. As the requirements and the technology supporting the architecture are updated every other day, it can be a hard task to keep up with the progress.

The overall success of designing depends on the fact that the scope of a project is planned well. In general, the initial scope of a project needs to be crystal clear, and it is very important to define it first.

The second goal for this project, to plan Metropolia Education Cloud Platform with SUSE OpenStack Cloud technology, was reached. However, several problematic obstacles were discovered during the designing phase. These were related to integration with a previously purchased hardware technology. After these were resolved, a reference architecture was generated for the use of Metropolia.

There is a number of topics that would have been useful to research more in relation to this project, such as how to design a PaaS or Containers as a Service (CaaS) layer to top of the Metropolia Education Cloud Platform. After this, the platform could also offer additional value-added services.

References

1 Bond J. The Enterprise Cloud. Sebastopol, CA: O'Reilly Media; 2015.

2 Giannakouris K, Smihily M. Cloud Computing - Statistics on the Use by Enterprises [online]. Eurostat; 28 February 2017.
URL: http://ec.europa.eu/eurostat/statistics-explained/index.php/Cloud_computing_-_statistics_on_the_use_by_enterprises.
Accessed on 24 February 2017.

3 Taylor C. Private Cloud Research 2015. Abergavenny, UK: Dynamic Markets Limited; 2015.

4 Adkins S, Belamaric J, Giersch V, Makogon D, Robinson J. OpenStack Cloud Application Development. Indianapolis, IN: John Wiley & Sons, Inc.; 2016.

5 OpenStack Foundation. OpenStack: a Business Perspective [online]. Austin, TX: OpenStack Foundation; September 2016.
URL: https://www.openstack.org/assets/pdf-downloads/business-perspectives.pdf.
Accessed on 24 February 2017.

6 Miller P, Nelson L, Liu F, Vargas S. OpenStack's Global Traction Expands for Its Newton Release. Cambridge, MA: Forrester Research, INC.; 19 December 2016.

7 OpenStack Foundation. What Is OpenStack [online]. Austin, TX: OpenStack Foundation; January 2016.
URL: http://www.openstack.org/software/. Accessed 20 January 2016.

8 Grance T, Mell P. The NIST Definition of Cloud Computing [online]. Gathersburg, MD: National Institute of Standards and Technology; September 2011.
URL: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf. Accessed 20 January 2016.

9 OpenStack Foundation. Accelerating NFV Delivery with OpenStack. Austin, TX: OpenStack Foundation; 1 March 2016.
URL: http://www.openstack.org/assets/telecoms-and-nfv/OpenStack-Foundation-NFV-Report.pdf. Accessed 20 January 2016.

10 SUSE. Datalounges Success Story [online]. Nuremberg, DE: SUSE; 22 September 2015.
URL: https://www.suse.com/docrep/documents/kgu61iyowz/datalounges_success_story.pdf. Accessed 6 February 2017

11 Nebula. Nebula Cloud 9 [online]. 2017.
URL: https://www.nebula.fi/en/ict-solutions/enterprise-ict/cloud-services/nebula-cloud-9. Accessed 6 February 2017.

12 Intel Corporation. Accelerating Business Growth with Industry-Standard, On-Premises IaaS [online]. Santa Clara, CA: Intel Corporation; June 2016
URL: http://www.intel.co.uk/content/www/uk/en/cloud-computing/bmw-uses-openstack-to-accelerate-business-growth-brief.html. Accessed 20 January 2017.

13 OpenStack Foundation. OpenStack: The Path to Cloud [online]. Austin, TX: OpenStack Foundation; 19 May 2016.
URL: https://www.openstack.org/assets/path-to-cloud/OpenStack-6x9Booklet-online.pdf. Accessed 6 February 2017.

14 Bumgardner C. OpenStack in Action. Shelter Island, NY. Manning Publication Co.; 2016.

15 OpenStack Foundation. OpenStack Releases [online]. Austin, TX: OpenStack Foundation; 23 March 2017.
URL: http://releases.openstack.org/. Accessed 26 March 2017.

16 OpenStack Foundation. Stable Branches [online]. Austin, TX: OpenStack Foundation; 13 June 2016.
URL: http://docs.openstack.org/project-team-guide/stable-branches.html. Accessed 26 March 2017.

17 OpenStack Foundation. Release Naming [online]. Austin, TX: OpenStack Foundation; 6 March 2017.
URL:https://wiki.openstack.org/wiki/Release_Naming. Accessed 20 January 2016.

18 Nelson L. OpenStack Is Ready - Are You?. Cambridge, MA: Forrester Research, INC; 18 May 2015.

19 SUSE. SUSE Openstack Cloud Reference Architecture with Dell Hardware [online]. Nuremberg, DE: SUSE; 16 October 2015
URL: https://www.suse.com/docrep/documents/r0ujxti4h7/suse_openstack_cloud_reference_architecture_with_dell_hardware_white_paper.pdf. Accessed 26 March 2017.

20 Sarat S, Shrivastwa A. Learning OpenStack. Brimingham, UK: Packt Publishing Ltd.; November 2015.

21 OpenStack Foundation. Picture of OpenStack Logical Architecture [online]. Austin, TX: OpenStack Foundation; Updated 16 Nov 2016.
URL: http://docs.openstack.org/ops-guide/_images/osog_0001.png. Accessed 16 November 2016.

22 Modeira B. CERN Cloud Architecture – February, 2016 [online].
URL: http://www.slideshare.net/moreirabelmiro/cern-cloud-architecture.
Accessed 6 February 2016.

23 OpenStack Foundation. Introduction to OpenStack High Availability [online]. Austin, TX: OpenStack Foundation; 26 March 2017.
URL: http://docs.openstack.org/ha-guide/intro-ha.html. Accessed 26 March 2017.

24 OpenStack Foundation. Configuring the Compute Node [online]. Austin, TX: OpenStack Foundation; 26 March 2017.
URL: https://docs.openstack.org/ha-guide/compute-node-ha.html. Accessed 26 March 2017.

25 Roth T, Sundermeyer S. SUSE OpenStack Cloud 6 Deployment Guide [online]. Nuremberg, DE: SUSE; 8 February 2017.
URL: https://www.suse.com/documentation/suse-openstack-cloud-6/singlehtml/book_cloud_deploy/book_cloud_deploy.html.
Accessed 26 March 2017.

26 SUSE. SUSE OpenStack Cloud 6 Release Notes [online]. Nuremberg, DE: SUSE; 3 March 2016.
URL: https://www.suse.com/releasenotes/x86_64/SUSE-OPENSTACK-CLOUD/6/. Accessed 20 January 2016.

27 OpenStack Foundation. Technical Considerations [online]. Austin, TX: OpenStack Foundation; January 2016.
URL: http://docs.openstack.org/arch-design/generalpurpose-technical-considerations.html. Accessed 20 January 2016.

28 Wikipedia. Cloud Computing Issues [online]. Wikipedia; 13 March 2017. URL: https://en.wikipedia.org/wiki/Cloud_computing_issues. Accessed 10 April 2016.

29 Wikipedia. x86 Virtualization [online]. Wikipedia; 23 March 2017. URL: https://en.wikipedia.org/wiki/X86_virtualization. Accessed 26 March 2017.

30 OpenStack Foundation. OpenStack Operations Guide [online]. Austin, TX: OpenStack Foundation; April 2016. URL: http://docs.openstack.org/openstack-ops/openstack-ops-manual.pdf. Accessed 20 April 2016.

31 OpenStack Foundation. Operational Considerations [online]. Austin, TX: OpenStack Foundation; January 2016. URL: http://docs.openstack.org/arch-design/generalpurpose-operational-considerations.html. Accessed 20 January 2016.

32 OpenStack Foundation. OpenStack User Survey [online]. Austin, TX: OpenStack Foundation; April 2016. URL: https://www.openstack.org/assets/survey/April-2016-User-Survey-Report.pdf. Accessed 26 March 2017.

33 OpenStack Foundation. Compliance Activities [online]. Austin, TX: OpenStack Foundation; February 2017. URL: http://docs.openstack.org/security-guide/compliance/compliance-activities.html. Accessed 6 February 2017.

34 Cisco Systems, Inc. SUSE Cloud Integration with Cisco UCS and Cisco Nexus Platforms [online]. San Jose, CA: Cisco Systems, Inc.; 2014 URL: http://www.cisco.com/c/dam/en_us/solutions/openstack/docs/cisco_and_suse.pdf. Accessed 11 February 2017.

Metropolia Education Cloud Platform: Technical Information

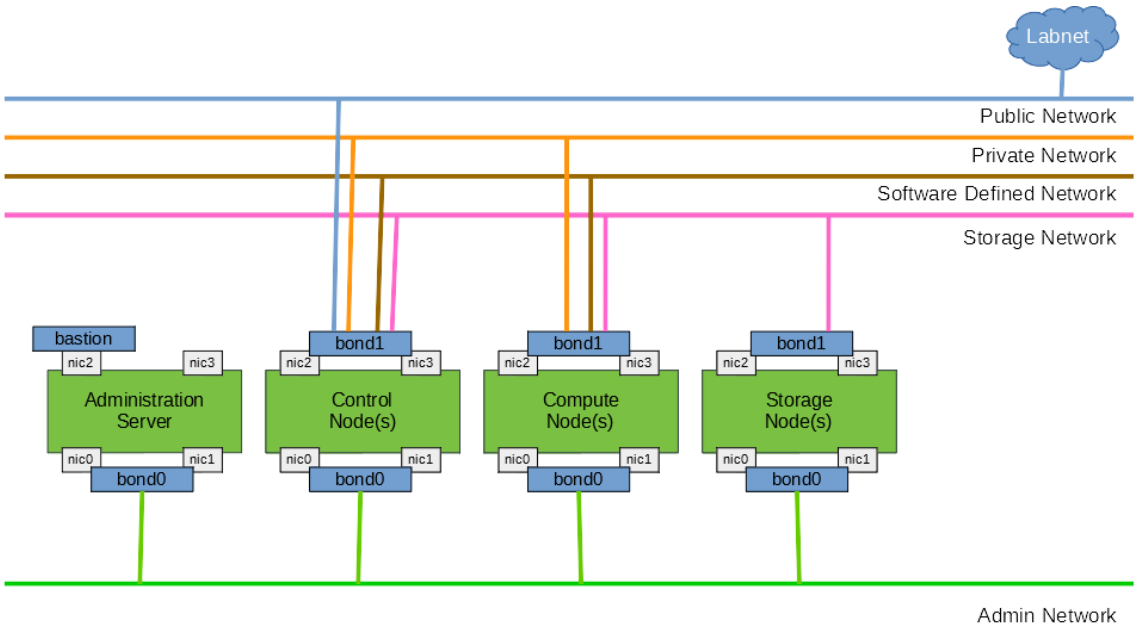
Hardware in use:

Hardware	Amount	Memory	Processors	Cores	Threads
Cisco UCS B200 M3	3	64 GB	2	16	32
Cisco UCS B200 M3	1	128 GB	2	16	32
Cisco UCS B200 M3	4	256 GB	2	16	32
Cisco UCS B200 M4	6	256 GB	2	20	40

SUSE OpenStack Cloud setup:

pcs	Function	Processors	Memory	Disk(s)	NIC(s)	Notes
1	Administration Server	2	64 GB	500 GB	4	
2	Control Node(s)	2	64 GB	500 GB + 50 GB	4	50 GB used for DRBD
10	Compute Node(s)	2	256 GB	300 GB + 1 TB	4	1TB used for ephemeral disks
1	Storage Node(s)	2	128 GB	300 GB + LUN	4	EMC providing LUN for Cinder RAW devices

Setup diagram:



SUSE OpenStack Cloud: Default Network Ranges

Name	Admin	Public Network	Private Network	Software Defined Network	Storage Network
Network	192.168.124.0	192.168.126.0	192.168.123.0	192.168.130.0	192.168.125.0
Netmask	/24	/24	/24	/24	/24
VLAN	Not supported	300	500	400	200
Purpose	A private network to access the Administration Server and all nodes for administration purposes. The default setup lets you also access the BMC (Baseboard Management Controller) data via IPMI (Intelligent Platform Management Interface) from this network. If required, BMC access can be swapped to a separate network.	The only public network provided by SUSE OpenStack Cloud. You can access the Nova Dashboard and all instances (provided they have been equipped with a floating IP) on this network. This network can only be accessed via a gateway, which needs to be provided externally. All SUSE OpenStack Cloud users and administrators need to be able to access the public network.	Private, SUSE OpenStack Cloud internal virtual network. This network is used for inter-instance communication and provides access to the outside world for the instances. The gateway required is also automatically provided by SUSE OpenStack Cloud	Private, SUSE OpenStack Cloud internal virtual network. This network is used when Neutron is configured to use openvswitch with GRE tunneling for the virtual networks. It should not be accessed by users	Private, SUSE OpenStack Cloud internal virtual network. This network is used by Ceph and Swift only. It should not be accessed by users.
Notes	router = 192.168.124.1 admin = 192.168.124.10 DHCP = 192.168.124.21 - 80 host = 192.168.124.81 - 160 bmc vlan host = 192.168.124.161 bmc host = 192.168.124.162 - 240 switch = 192.168.124.241 - 250 BMC VLAN (bmc_vlan) 100	router = 192.168.126.1 public host = 192.168.126.2 - 127 floating host = 192.168.126.129 - 254	DHCP = 192.168.123.1 - 254	host = 192.168.130.10 - 254	host = 192.168.125.10 - 239